SW Engineering CSC648/848 Section 01 Fall 2017
Bay Real Estate, Team 12 Local, Milestone 4
December 5th, 2017

Jason Cromer (jcromer@mail.sfsu.edu)
Artsem Holdvekht
Chen-Feng Huang
David Hoang
Jiawen Zhu
Jordan Leong

Date Submitted	Date Revised and Frozen
12/08/17	

Product Summary

- Name of product: Real estate demo
- All major committed functions:
  - Our website displays featured **listings** and/or new **listings**.
  - Clicking on a picture will zoom-in on it and show the detailed description of the listing.
  - **All Users** shall be able to **browse** all listings on the home page.
  - **All Users** shall be able to **search** listings by street address, city, state, and/or Zip.
  - **All Users** shall **sign in** to the website in order to access seller contact information.
  - **Registered Buyers** shall be prompted to register their **name**, **phone number**, and optional **email**.
  - **Registered Sellers** shall be prompted to register their **name**, **email**, **phone number**, and **license** number for verification.
  - **Registered Sellers** shall be able to list and edit property information such as **description of house**, and **photos**.
- URL of our website: https://sfsuse.com/fa17g12/

Usability Test Plan

Test objectives:

Test/use case	% completed	errors	comments
search	100%	Has not found yet	The search bar has been tested with Street address, city name and zip code

Test Plan:

1) System setup:

Using express node.js as the back end framework.

2) Starting point: List all the tokens that user could search in the search bar such as city name, and street address, zip code…. Take invalid token as mistyping. Handel that in HTML in efficient way.

3) Task to be accomplished:

Type Either street address or city name or zip code in the search bar or all of three at once.
The result will show the **House image**, **street address**, **city name**, **zip code** and according **state**.

4) Intended user:

potential house buyer: who are interested buying a house or hunting a house to buy in a future.
real estate agent: who have real estate agent license. And the person who is welcome to post as many houses as the person wants to.

5) Completion criteria: The search engine will take user request from the search bar and find the closest result that match the user needs.

6) URL of the system to be tested: https://sfsuse.com/fa17g12/

Questionnaire form

Questionnaire	Strong agree	Agree	Neutral	Disagree	Strong disagree
The efficiency of using search bar to look for some in the listing					
User can easily find any tag, ant button, or any listing they want to find					
Every function shows easy and clear instruction					
The description of all listing show enough information to users					

QA Test Plan

This test plan has been created in order to make clear the objectives and details of the search to others.

3.1 Test objectives

The objective of this test is to verify that the search for listings functions according to specifications, and to check for possible errors.

3.2 Hardware and software setup

Hardware – Any computer with an internet connection.
Software – A web browser such as Chrome or Firefox.
The search uses Express and Javascript to query listings' information from a MySQL database.

3.3 Feature to be tested

The search feature on the front page of the website will be tested for different types of inputs of varying lengths, which should yield the expected results. Each test will be repeated twice to check for consistency of each result; one test will be done in Chrome, and an identical test will be done in FireFox.
The search function is intentionally slightly vague in order to try and recommend other possible listings that the user may want to see.
The search bar should also retain the user's input after searching.

3.4 Actual test cases

Test #	Test Title	Description	Input	Expected Output	Test Result
1	Chrome Search Lower-Case	Searching by city in lower-case.	san francisco	One result: 431 25th Ave.	PASS
2	Firefox Search Lower-Case	Searching by city in lower-case.	san francisco	One result: 431 25th Ave.	PASS
3	Chrome Search Upper-Case	Searching by city in upper-case.	SAN FRANCISCO	One result: 431 25th Ave.	PASS
4	Firefox Search Upper-Case	Searching by city in upper-case.	SAN FRANCISCO	One result: 431 25th Ave.	PASS
5	Chrome No Input	Attempting search without an input.		Display: “Please enter a query.”	PASS
6	Firefox No Input	Attempting search without an input.		Display:: “Please enter a query.”	PASS
7	Chrome Gibberish Input	Special characters and nonsense.	-*$%5-&2JG H*$IN()$5	Display: “No data”	PASS
8	Firefox Gibberish Input	Special characters and nonsense.	-*$%5-&2JG H*$IN()$5	Display: “No data”	PASS
9	Chrome Zip Code Search	Searching by zip code.	90069	One result: 1635 Woods Dr.	PASS
10	Firefox Zip Code Search	Searching by zip code.	90069	One result: 1635 Woods Dr.	PASS

Code Review

The coding style we used included separate files one for display, one for scripts to apply on the display, and one for the actual search. Below shows the search function.

Hey Jordan,
Could you take a look at the search function I did? At the moment it searches for the first three characters in the search input. Anything I should change?

```
 var searchListing = function (req,res,next){
       var listing = req.params.listing;
       req.getConnection(function(err,conn)
       {
          if (err) return next("Cannot Connect");
          var query = conn.query("SELECT * FROM listings WHERE city LIKE ? ", listing[0] +
listing[1] +  listing[2] + "%", function(err,rows){
             if(err){
                console.log(err);
                return next("Mysql error, check your query");
             }
             //if listing not found
             if(rows.length < 1)
                return res.send("Listing not found.");
             res.render('index',{data:rows});
          });
       });
};
```

Thanks,
David

Hey David, thanks for asking. Anyways regarding your code as of right now it seems to do a %LIKE search using specifically 3 or less characters and also we need to be able to search ZIP or address. Perhaps we could do two methods of searching in which we either use 4 chars or less OR if 5or more chars we search with that as follows:

```
        //like search 1-4 characters
        var searchWord = "%";
         if (listing.length > 0 && listing.length < 5){
                for(i = 0; i < listing.length; i++){
                     searchWord = searchWord + listing[i]
                 };
         }
         else{//like search maximum 5 characters
                  searchWord = "%" + listing[0] + listing[1] + listing[2] + listing[3] + listing[4] +
"%";
         }
        var query ="SELECT * FROM listings WHERE city LIKE ? OR address LIKE ? OR
zip_code LIKE ? "
```

-Best Regards,
Jordan

Ok, here are my changes according to your feedback.

```
var searchListing = function (req, res, next) {
   var listing = req.params.listing;
   var returning = false;
   //like search 1-4 characters
  var searchWord = "%";
  if (listing.length > 0 && listing.length < 5){
          for(i = 0; i < listing.length; i++){
                searchWord = searchWord + listing[i];
       }
 }
 else{
          //like search maximum 5 characters
          searchWord = "%" + listing[0] + listing[1] + listing[2] + listing[3] + listing[4] + "%";
 }
 searchWord = searchWord + "%";
  var queryString = "SELECT * FROM listings WHERE city LIKE ? OR address LIKE ? OR
zip_code LIKE ? "
   var filter = [searchWord, searchWord, searchWord];
    var query = dbConnection.query(queryString, filter, function (err, rows) {
          if (err) {
                console.log(err);
                return next("Mysql error, check your query");
           }
           //if listing not found
            if (rows.length < 1){
```

```
            //return res.send("Listing not found.");
            }
            res.render('index', {data: rows, returning: returning});
});
};
```

Best practices for security

Considering our business is built on service that requires exchange of personal information, specifically between the seller and potential real estate buyer. Even though we are not storing very sensitive personal information such as Social Security Number(SSN), or Tax Information, we very concerned with the security of our service and the ability to protect customers authentication information. The biggest concern are the customer's passwords. In order to protect and safely store users' passwords we have employed a Bcrypt technology. It is based on the generating a cryptographically strong hashes of user passwords over the course of a number of hashing rounds. And performing a comparisons of these hashes during the authentication. That way the passwords are saved in unreadable form so the perpetrators won't be able to decipher the password if stolen.

In order to be able to acquire more accurate information from users during registration and further usage of functions such as adding/editing listings we employed field validation. At the moment the validation is only performed on the back end of the server, but we are planning to expand it to the front end as well, that essentially doubling the security. At the moment we are lacking the ability to verify the actual zip code, for example, so in order to ensure some what accurate input we are checking that the input is numeric and six digits long., similar for the phone numbers. As far as the names of the customers and we are unable to establish the credibility of the input, so we are using our admin to at least ensure that the names and any text information is not offensive or disturbing. If the requirements are not satisfied user will have receive message describing error, and will be asked to fix/enter missing fields. On contrary if our admins will find material that is not suitable he/she reserves to remove that entry or user.

Non-Functional Specs Adherence

1. Application shall be developed and deployed using class provided deployment stack (DONE)
2. Application shall be developed using pre-approved set of SW development and collaborative tools provided in the class. Any other tools or frameworks must be explicitly approved by Anthony Souza on a case by case basis. (DONE)
3. Application shall be hosted and deployed on Amazon Web Services as specified in the class (DONE)
4. Application shall be optimized for standard desktop/laptop browsers e.g. must render correctly on the two latest versions of all major browsers: Mozilla, Safari, Chrome. (ON TRACK)
5. Application shall have responsive UI code so it can be adequately rendered on mobile devices but no mobile native app is to be developed (ON TRACK)
6. Data shall be stored in the MySQL database on the class server in the team's account (DONE)
7. Application shall provide real-estate images and optionally video (DONE)
8. Maps showing real-estate location shall be required (DONE)
9. Application shall be deployed from the team's account on AWS (DONE)
10. No more than 50 concurrent users shall be accessing the application at any time (ON TRACK)
11. Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users. (DONE)
12. The language used shall be English. (DONE)
13. Application shall be very easy to use and intuitive. No prior training shall be required to use the website. (DONE)
14. Google analytics shall be added (ON TRACK)
15. Messaging between users shall be done only by class approved methods and not via e-mail clients in order to avoid issues of security with e-mail services. (DONE)
16. Pay functionality (how to pay for goods and services) shall not be implemented. (DONE)
17. Site security: basic best practices shall be applied (as covered in the class) (DONE)
18. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development (DONE)
19. The website shall prominently display the following text on all pages "SFSU Software Engineering Project, Fall 2017. For Demonstration Only". (Important so as to not confuse this with a real application). (ON TRACK)